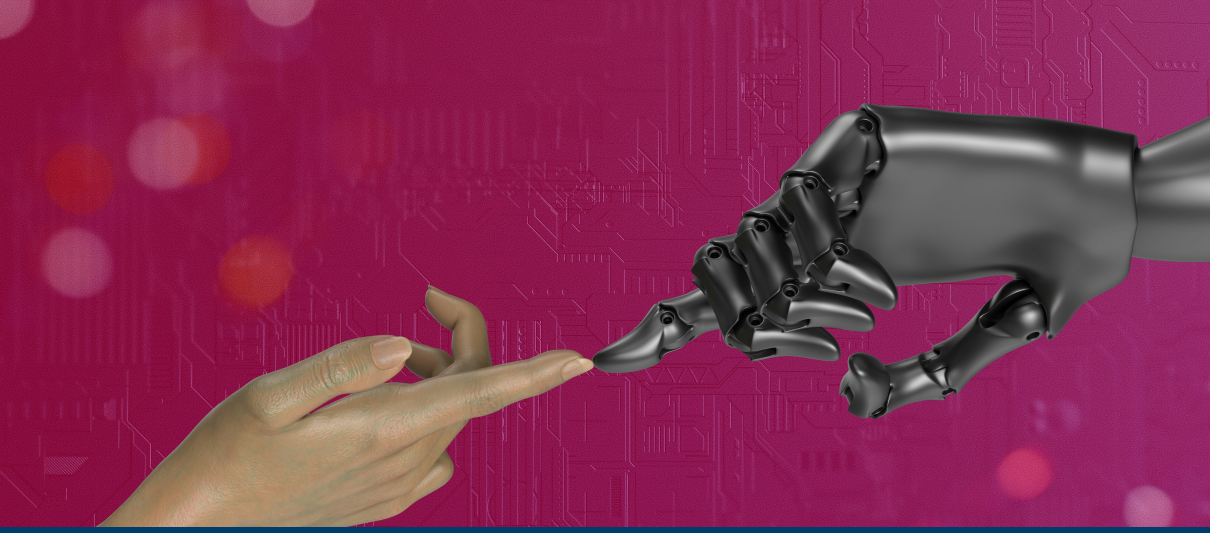


Hands-On Machine Learning

Dr. Derek Bridge | 2026/2027

UCC Coláiste na hOllscoile Corcaigh
University College Cork, Ireland

School of Computer Science and Information Technology
Scoil na Ríomheolaíochta agus na Teicneolaíochta Faisnéise

7. Linear Regression

In this lecture, we use the same housing data that we used in two of the previous lectures.

Parametric learning

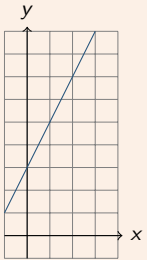
- In **parametric learning**, the model that we will learn has a structure that is known in advance — a function of a particular form.
- But the function contains **parameters** (numeric variables).
- During training, it is the job of the learning algorithm to choose values for the parameters.

Examples

- Parametric learning: Linear Regression, Logistic Regression, most Neural Networks.
- Non-parametric learning: Decision Trees, kNN.

Equation of a Straight Line

- E.g. $y = 3 + 2x$
- Question: From the point of view of plotting this line, what's 3? What's 2?
- In general, $y = b + wx$
- In maths, b and w are called **coefficients**.
- In ML, they are the **parameters**.
- In ML, b might be called the **bias** and w is a **weight**.



Linear Equations

In general, a **linear equation** is of the form:

$$\begin{aligned} y &= b + \boldsymbol{w}_1\boldsymbol{x}_1 + \cdots + \boldsymbol{w}_n\boldsymbol{x}_n \\ &= b + \sum_{i=1}^{n} \boldsymbol{w}_i\boldsymbol{x}_i \\ &= \boldsymbol{wx} \text{ (explained on next two slides)} \end{aligned}$$

Vectors

- In HOML, an n-dimensional **vector** is just an array of n numbers, referred to as the **elements** of the vector.
- We use bold, lowercase letters for vectors, e.g.

$$\boldsymbol{x} = [2, 4, 3]$$

Dot Product

- Suppose we have two n-dimensional vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$.
- Their **dot product**, $\boldsymbol{uv}$, is computed by summing the products of corresponding elements.

$$\begin{aligned}\boldsymbol{uv} &= \boldsymbol{u}_1\boldsymbol{v}_1 + \boldsymbol{u}_2\boldsymbol{v}_2 + \cdots + \boldsymbol{u}_n\boldsymbol{v}_n \\ &= \sum_{i=1}^{n} \boldsymbol{u}_i\boldsymbol{v}_i\end{aligned}$$

Exercise

- Let $\boldsymbol{u} = [2, 4, 3]$ and $\boldsymbol{v} = [5, 0, 1]$.
- Compute $\boldsymbol{uv}$.

Writing a Linear Equation as the Dot Product of Two Vectors

- Suppose we have a **linear equation**

$$y = b + \boldsymbol{w}_1\boldsymbol{x}_1 + \cdots + \boldsymbol{w}_n\boldsymbol{x}_n$$

- Collect all the $\boldsymbol{x}$ values into an n-dimensional vector:

$$\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$$

  Collect all the weights into an n-dimensional vector:

$$\boldsymbol{w} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n]$$

  Then the linear equation can be written as a dot product:

$$y = b + \boldsymbol{w}\boldsymbol{x}$$

- Better, insert a cheeky 1 into vector $\boldsymbol{x}$ (so that it is now $n+1$-dimensional):

$$\boldsymbol{x} = [1, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]$$

  Insert b into vector $\boldsymbol{w}$ (so it too is $n+1$-dimensional):

$$\boldsymbol{w} = [b, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n]$$

  Then the linear equation can be written as this dot product:

$$y = \boldsymbol{w}\boldsymbol{x}$$

Linear Regression with one feature

- If there's just one feature $\boldsymbol{x}_1$, the model we learn will be of the form $\hat{y} = b + \boldsymbol{w}_1 \boldsymbol{x}_1$
- In training, the goal is to find values for the parameters b and $\boldsymbol{w}_1$.
- In other words, we're fitting a *line* to the training data: the line that 'goes through the middle of the data'.

Linear Regression with two features

- If there are two features $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, the model we learn will be of the form $\hat{y} = b + \boldsymbol{w}_1 \boldsymbol{x}_1 + \boldsymbol{w}_2 \boldsymbol{x}_2$
- In training, the goal is to find values for the parameters b, $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$.
- In other words, we're fitting a *plane* to the training data, again 'going through the middle of the data'.

Linear Regression with more than two features

- If there are n features $\boldsymbol{x}_1 \dots \boldsymbol{x}_n$, the model we learn will be of the form $\hat{y} = b + \boldsymbol{w}_1 \boldsymbol{x}_1 + \cdots \boldsymbol{w}_n \boldsymbol{x}_n$
- In training, the goal is to find values for the parameters b, $\boldsymbol{w}_1 \dots \boldsymbol{w}_n$.
- In other words, we're fitting a *hyperplane* to the training data, again 'going through the middle of the data'.

Why Linear Models?

Real-world problems are not usually linear but they might be nearly linear.

- Training is fast; inference is fast.
- A good baseline: if a fancy model's error is not significantly better, then maybe it's not worth the bother.

- Suppose we have n features. Then there are $n + 1$ parameters. Why?
- For concreteness, assume two features, hence three parameters.
- During training, Linear Regression must find the best values for these parameters. Let's refer to each choice as a **hypothesis**, h.
- E.g. one hypothesis: $b = 800$, $\boldsymbol{w}_1 = -5$, $\boldsymbol{w}_2 = 3.7$.
- E.g. another hypothesis: $b = -10.34$, $\boldsymbol{w}_1 = 0$, $\boldsymbol{w}_2 = 3.1459$
- Which of these is better? Which 'goes through the training data' best?
- We need to *score* the hypotheses so we can pick the one with the best score.
- The score of each hypothesis is computed by a **loss function**.

Loss Function

- The loss function, designated J, takes in a hypothesis h and says how good it is.
- For each example $\boldsymbol{x}$ in the training set, it computes the difference between h's predicted value for $\boldsymbol{x}$ (i.e. $h(\boldsymbol{x})$) and $\boldsymbol{x}$'s actual value, y.
- N.B. Low scores are better!

Ordinary Least-Squares Regression (OLS)

- The most common loss function for regression is the mean squared error (MSE):

$$J(\boldsymbol{X}, \boldsymbol{y}, h) = \frac{1}{m} \sum_{\boldsymbol{x} \in \boldsymbol{X}, y \in \boldsymbol{y}} (h(\boldsymbol{x}) - y)^2$$

  where $\boldsymbol{X}$ is the training set with $\boldsymbol{y}$ its labels.
- Linear Regression using this loss function is referred to as **Ordinary Least-Squares Regression (OLS)**:
  - "least" because we are minimizing the loss;
  - "squares" because the loss is mean squared error;
  - "ordinary" to distinguish it from more exotic variants.
- Question: why are we squaring? (Two answers.)
- In fact, we often divide by 2:

$$J(\boldsymbol{X}, \boldsymbol{y}, h) = \frac{1}{2m} \sum_{\boldsymbol{x} \in \boldsymbol{X}, y \in \boldsymbol{y}} (h(\boldsymbol{x}) - y)^2$$

Why use MSE for the Loss Function?

- MSE is continuously differentiable.
- MSE is **convex**.
  - Informally, this means that it has a unique minimum.
  - When there are two features, loss is bowl-shaped.
  - (To be 100% accurate: there is a unique minimum provided no feature in $\boldsymbol{X}$ is linearly dependent on any of the others or, equivalently, provided $\boldsymbol{X}$ has an inverse.)

Finding the Best Hypothesis

Why does this not work?

for *each possible hypothesis* h **do**
    compute h's loss, $J(\boldsymbol{X}, \boldsymbol{y}, h)$;

end

return the h that had the lowest loss

Normal Equation

Given the training data $\boldsymbol{X}$ and $\boldsymbol{y}$, the **Normal Equation** computes the winning hypothesis:

$$(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Gradient Descent

start with an initial guess for the values of the parameters;

repeat
    update the guess, changing the parameter values in a way that reduces the loss;

until *convergence*;

Deriving the Normal Equation

- We need the **gradient** of the loss function with regard to each $\boldsymbol{w}_i$:

$$\frac{\partial J(\boldsymbol{X}, \boldsymbol{y}, h)}{\partial \boldsymbol{w}_i}$$

  - In other words, how much the loss will change if we change $\boldsymbol{w}_i$ a little.
  - With respect to a particular $\boldsymbol{w}_i$, it is called the **partial derivative**.
  - The **gradient vector** contains each of these partial derivatives.

$$\nabla J(\boldsymbol{X}, \boldsymbol{y}, h) = \begin{bmatrix} \frac{\partial J(\boldsymbol{X}, \boldsymbol{y}, h)}{\partial b} \\ \frac{\partial J(\boldsymbol{X}, \boldsymbol{y}, h)}{\partial \boldsymbol{w}_1} \\ \vdots \\ \frac{\partial J(\boldsymbol{X}, \boldsymbol{y}, h)}{\partial \boldsymbol{w}_n} \end{bmatrix} = \begin{bmatrix} \frac{1}{m} \sum_{\boldsymbol{x} \in \boldsymbol{X}} (b - y) \\ \frac{1}{m} \sum_{\boldsymbol{x} \in \boldsymbol{X}} (\boldsymbol{w}_1 \boldsymbol{x}_1 - y) \times \boldsymbol{x}_1 \\ \vdots \\ \frac{1}{m} \sum_{\boldsymbol{x} \in \boldsymbol{X}} (\boldsymbol{w}_n \boldsymbol{x}_n - y) \times \boldsymbol{x}_n \end{bmatrix}$$

- Set $\nabla J(\boldsymbol{X}, \boldsymbol{y}, h) = \boldsymbol{0}$.
- Do some algebraic manipulation to get $b, w_1 \dots w_n$ on the left-hand side, and we get

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$